APR 0 6 2004
PATENT & TRADEMARK OFFICE

Docket No. AT9-99-319 *PATENT*

#17
[illegible]
4-13-04

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Broussard**	§	
	§	Group Art Unit: **2155**
Serial No. **09/392,841**	§	
	§	Examiner: **Qureshi, Shabana**
Filed: **September 9, 1999**	§	
	§	
For: **Method and System for Remote Java Audio Server in a Heterogeneous Distributed Environment**	§	

RECEIVED
APR 0 9 2004
Technology Center 2100

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

Certificate of Mailing Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being deposited with the United States Postal Service as First Class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on April 2, 2004.

By: Michele Morrow
Michele Morrow

(972) 367-2001
x 114

APPELLANT'S BRIEF (37 C.F.R. 1.192)

This brief is in furtherance of the Notice of Appeal, filed in this case on February 5, 2004.

The fees required under § 1.17(c), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief is transmitted in triplicate. (37 C.F.R. 1.192(a))

04/08/2004 [illegible] 00000027 090447 09392841
01 FC:1402 330.00 DA

REAL PARTIES IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines, Inc.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-16

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: NONE
2. Claims withdrawn from consideration but not canceled: NONE
3. Claims pending: 1-16
4. Claims allowed: NONE
5. Claims rejected: 1-16

C. CLAIMS ON APPEAL

The claims on appeal are: 1-16

STATUS OF AMENDMENTS

There are no amendments after final rejection.

SUMMARY OF INVENTION

A method and system for an audio server in a heterogeneous distributed environment is provided. A Java application executes on a host machine under X Windows or RAWT (Remote Abstract Window Toolkit), and the Java application generates audio data and graphic data. The graphic data is sent to a display server on a client machine specified by a display environment variable. Although neither X Windows nor RAWT have audio support, a Java audio driver on the host machine determines whether an audio environment variable or an audio command line parameter is specified on the host machine. In parallel to the graphic data, the audio data is then sent to a Java audio server on the client machine specified by the audio environment variable or the audio command line parameter, and played using the local audio support, in Java, on the client machine on which the user can hear the audio.

ISSUES

The issues on appeal are whether claims 1-2, 4-6, 8-11 and 13-16 are unpatentable over Goetz et al. (U.S. Patent No. 5,928,330) in view of Slaughter et al. (U.S. Patent No. 6,202,147 B1) and whether claims 3, 7, 12 and 16 are unpatentable over Goetz and Slaughter and further in view of Semenzato (U.S. Patent No. 5,903,728).

GROUPING OF CLAIMS

The claims do not stand or fall together for the reasons set forth hereafter in Appellant's arguments. The claims stand or fall according to the following grouping of claims:

Group I:	claims 1 and 14;
Group II:	claims 2 and 15;
Group III:	claims 4 and 9;
Group IV:	claims 5 and 10.
Group V:	claims 6 and 11;
Group VI:	claims 8 and 13;
Group VII:	claims 3 and 16; and
Group VIII:	claims 7 and 12.

ARGUMENT

I. 35 U.S.C. 103(a), Alleged Obviousness, Claims 1-2, 4-6, 8-11 and 13-16

The Final Office Action rejects claims 1-2, 4-6, 8-11 and 13-16 under 35 U.S.C. 103(a) as being allegedly unpatentable over Goetz et al. (U.S. Patent No. 5,928,330) in view of Slaughter et al. (U.S. Patent No. 6,202,147 B1). This rejection is respectfully traversed.

With regard to claim 1, the Final Office Action states:

> Regarding claim 1, Goetz taught a method for a distributed audio server (abstract), the method comprising the computer implemented steps of:
> Generating audio data and graphic data in a platform-independent application (column 9, line 31-35, column 11, lines 2-8);
> Sending the graphic data to a display server on a client machine specified by a display environment variable (column 11, lines 15-48, column 1, lines 61-65); and
> Sending the audio data to an audio server on the client machine specified by an audio environment variable or an audio command line parameter (column 11, lines 15-48, column 1, lines 61-65).
> Goetz does not specifically teach the audio server is a platform-independent audio server. However, Slaughter taught a platform-independent device server (column 3, line 66 – column 4, line 8). It would have been obvious to one of ordinary skill in the art at the time the invention was made that incorporating Slaughter's platform independent device server in Goetz's distributed multimedia system would have improved system effectiveness. The motivation would have been to provide even greater support for the diverse capabilities associated with the different available platforms.

Final Office Action dated January 14, 2004, pages 2-3.

Independent claim 1, which is representative of independent claim 14 with regard to similarly recited subject matter, reads as follows:

> 1. A method for a distributed audio server, the method comprising the computer-implemented steps of:
> generating audio data and graphic data in a platform-independent application;
> <u>sending the graphic data to a display server on a client machine specified by a display environment variable</u>; and
> <u>sending the audio data to a platform-independent audio server on the client machine specified by an audio environment variable or by an audio command line parameter</u>. (emphasis added)

Appellant respectfully submits that neither Goetz nor Slaughter teach or suggest the features emphasized above. Goetz teaches a method of presenting multimedia information on a presentation device. The system of Goetz assigns an importance value to each unit of multimedia information, which is indicative of the quality of presentation, and gathers performance capability of the system. The characterized performance capability is analyzed to infer network conditions. Thus, the server may stream at a streaming rate to adapt to the importance of the information and inferred network conditions from the analysis (Abstract). However, Goetz does not teach sending the graphic data to a display server on a client machine specified by a display environment variable. Goetz also does not teach sending the audio data to a platform-independent audio server on the client machine specified by an audio environment variable or by an audio command line parameter.

The Final Office Action alleges that these features are taught at column 11, lines 27-34 and column 1, lines 61-65 of Goetz, which reads as follows:

> To view a file, the web browser requests a Universal Resource Locator (URL) from the web server application, and the web server application responds with a message which includes a MIME type which specifies the location and type of the file. Based on the file type, the web browser application may need to invoke a "helper" application, which is used specifically to handle files having the specified file type. In the case of multimedia file **850**, the web browser application invokes the novel multimedia client application, which initiates an interaction with the novel multimedia server application to produce the multimedia file.
>
> An exemplary initial interaction between the multimedia client application **1020** and the multimedia server application **1040** is described below with reference to **FIG. 11**. The interaction begins in step **1100** and proceeds to step **1110**, where the multimedia client application **1020** sends a media request message to the multimedia server application **1040** specifying the desired media types to be produced, and specifying a version number of the multimedia client application and a port number to which the multimedia server application should direct communication. In step **1120**, the multimedia client application **1020** sends a message to the multimedia server application **1040** specifying a desired rate of transmission. The desired rate of transmission may be determined by the client, for example, by determining the communication rate of an attached communication device such as a modem (more below). The interaction then proceeds to step **1130**, where the multimedia client application **1020** sends a "go" message to the server application **1040**, informing the multimedia server application that the initial client messages have been sent, and therefore allowing the server to determine whether or not all messages were received.

(Column 11, lines 15-48)

> To provide intelligent audio-video clips, the video data must be provided to a video driver and audio data must be provided to a sound card driver within specified timing tolerances to maintain intra- and inter-stream synchronism.

(Column 1, lines 61-65)

In the above sections, Goetz teaches producing media files by initiating a multimedia client application that interacts with a multimedia server application. The multimedia client application first sends a media request message to the multimedia server application specifying the desired media type, the version number of the multimedia client application, and the port number to which the multimedia server application directs communication. The multimedia client application also sends a desired rate of transmission as determined by the client. Once the initial client messages have been sent, the multimedia client application sends a "go" message to the multimedia server application in order for the multimedia server application to determine whether or not all messages sent were received.

While Goetz teaches, in the above section, generating media files using a multimedia server application, Goetz does not teach that the multimedia server application sends graphic data to a display server on a client machine specified by a display environment variable or audio data to a platform-independent audio server on the client machine specified by an audio environment variable or by an audio command line parameter. Goetz only teaches sending messages between a single multimedia client application and a single multimedia server application. Goetz does not teach sending the graphic data to a display server on a client machine and the audio data to an audio server on the client machine. The display server and the audio server in the presently claimed invention are two different applications running on the client machine. To the contrary, Goetz only teaches a single multimedia client application that requests media files to be generated by the multimedia server application.

In addition, since Goetz only teaches a single multimedia client application instead of a display server and an audio server, which are two different applications, Goetz does not, and would not, teach a display environment variable that specifies a display server and an audio environment variable/audio command line parameter that specifies an audio server. There is no mention of a display environment variable or an audio environment variable/audio command line parameter anywhere in the reference. Goetz only specifies a version and a port number of a single multimedia client application that receives media files in a message, Goetz does not specify an audio environment variable for an audio server and a display environment variable for a display server, which are two separate variables used for two different applications running on the client machine. Therefore, Goetz does not teach a display server on a client machine specified by a display environment variable or an audio server on the client machine specified by an audio environment variable or an audio command line parameter, as recited in claim 1.

The Final Office Action admits that Goetz does not teach that an audio server is a platform-independent audio server. However, the Final Office Action alleges that Slaughter teaches a platform-independent audio server at column 3, line 66 to column 4, line 8, which reads as follows:

> Computer system **100** includes a central processing unit ("CPU") **102**, for example, a Sun Microsystems SPARC, Motorola PowerPC, or Intel Pentium processor, CPU **102** is coupled in turn with memory **104**. Memory **104** can include any type of memory device used in a computer system, including random access memory ("RAM") and read-only memory ("ROM"). CPU **102** is also coupled with a bus **106**, such as a PCI bus, or an S bus. A variety of input devices **108** and **110**, and output devices **112** are also coupled with bus **106**.

Appellant respectfully disagrees. In the above section, Slaughter only teaches a computer system that includes a processor that is coupled with memory and other memory devices, such as a random access memory. The processor also couples with a bus, which may be coupled with other input devices. Slaughter does not teach a platform-independent audio server, as recited in claim 1.

In the reference, Slaughter teaches a platform-independent device driver that obtains memory by requesting memory objects from a bus manager, which then allocates a real memory object, which has real system memory resources. Thus, the device driver requires no knowledge of the platform on which it operates other than knowledge of the bus architecture used by the platform (Column 3, lines 43-62). However, the device driver of Slaughter is not the same as the audio server, as recited in claim 1.

The device driver of Slaughter requests an allocation of memory from the bus manager. The audio server of the presently claimed invention is specified by an audio environment variable or audio command line parameter to receive audio data sent from a host. Therefore, one of ordinary skill in the art would not be led to modify the device driver of Slaughter to include Goetz's multimedia client application to reach the presently claimed invention, because the device driver of Slaughter is non-analogous to the audio server. The device driver of Slaughter has nothing to do with sending or receiving audio data. The device driver merely requests allocation of memory from the bus manager. Therefore, a person of ordinary skill in the art would not take a platform-independent device driver that is used to request allocation of memory and apply it to receive audio data, without the previous disclosure of the Appellant.

The Final Office Action further alleges that it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate Slaughter's platform-independent device driver in Goetz's distributed multimedia system because it would have improved system effectiveness. The alleged motivation is to provide even greater support for the diverse capabilities associated with the different available platforms. There is no teaching or suggestion in either of Goetz or Slaughter to make the combination alleged by the Final Office Action. Goetz is only concerned with using multimedia client and server applications to exchange messages to request a media file, so that a user of a Web browser may view Web pages with media files stored on a server. Goetz does not suggest that a need exists for a platform-independent multimedia client application. Slaughter is only concerned with using a platform-independent device driver to request allocation of memory, so that the costs of device driver development for different platforms are reduced. Therefore, the motivation of "providing greater support for diverse capabilities with different platforms" is simply not supported in either of Goetz or Slaughter. One of ordinary skill in the art, being presented with only Goetz and Slaughter and not having any prior knowledge of Appellant's claimed invention, would not have found any reason to combine the references and modify them in the particular manner that would be necessary to arrive at sending audio data to a platform-independent audio server running on the client machine specified by an audio environment variable or an audio command line parameter.

Furthermore, even if a person of ordinary skill in the art were somehow motivated to combine or modify Goetz to include Slaughter's platform-independent device driver, the result would not be a platform-independent audio server, as recited in claim 1. The result would be a platform-independent device driver that requests allocation of memory for requesting audio data, not a platform-independent audio server that is specified by an audio command line parameter or an audio environment variable to receive audio data. Therefore, the alleged combination of the references still would not teach or suggest the features of sending graphic data to a display server on a client machine specified by a display environment variable, and sending the audio data to a platform-independent audio server on the client machine specified by an audio environment variable or by an audio command line parameter.

In view of the above, Appellant respectfully submits that neither Goetz nor Slaughter, either alone or in combination, teach or suggest the features recited in claim 1. In addition, independent claim 14 recites similar subject matter also not taught by Goetz or Slaughter. Thus, Appellant respectfully requests withdrawal of the rejection of claims 1 and 14. At least by virtue of their dependency on claims 1 and 14, respectively, neither Goetz nor Slaughter, either alone or in combination, teach or suggest the features set forth in dependent claims 2, 15, and 16. Accordingly, Appellant respectfully requests withdrawal of the rejection of claims 1-2 and 14-16 under U.S.C. 103(a).

With regard to dependent claim 2, which is representative of claim 15 with regard to similarly recited subject matter, neither Goetz nor Slaughter teach or suggest that the platform-independent application and the platform-independent audio server are implemented in the Java programming language. The Final Office Action alleges that Slaughter teaches these features at column 4, lines 48-53, which reads as follows:

> These applications run in conjunction with runtime system 208. Runtime system 208 includes, in one embodiment, a Java Virtual Machine ("JVM") 210 that receives instructions in the form of machine-independent bytecodes produced by the application running in applications layer 206 and interprets the instructions by converting and executing them.

In the above section, Slaughter teaches that application programs that are commonly run by computer users run in conjunction with the Java Virtual Machine. The Java Virtual Machine receives instructions from the application program, interprets the instructions and executes them. Thus, Slaughter teaches a platform independent application that may be implemented in Java programming language. However, Slaughter does not teach a platform-independent audio server that is implemented in the Java programming language. As described above, Slaughter only teaches a platform-independent device driver that requests allocation of memory, not a platform-independent audio server that receives audio data. Therefore, Slaughter does not, and would not, teach a platform-independent audio server that is implemented in Java programming language.

With regard to independent claim 4, which is representative of independent claim 9 with regard to similarly recited subject matter, neither Goetz nor Slaughter teach or suggest <u>determining whether an audio environment variable is defined or an audio command line parameter is defined</u> or <u>if an audio environment variable or an audio command line parameter is defined, sending the audio data to an audio server on a client machine specified by the audio environment variable or by the audio command line parameter</u>. The Final Office Action alleges that Goetz teaches these features at column 11, lines 15-48 and column 1, lines 61-65, which are reproduced above, and at column 12, lines 27-34, which reads as follows:

> After streaming a time-slice's worth of the packets in step **1220**, the logic proceeds to step **1230**, where the logic requests to go to sleep, or go idle, for a time quanta corresponding to the streaming rate. For example, if the logic is streaming audio to correspond to 960 ms slice of presentation, step **1230** will request to go to sleep until it is time to stream another 960 ms worthier of information.

In the above section, Goetz only teaches streaming logic that is necessary to continue streaming data by a time-slice's worth of packets. The logic then requests to go to sleep, or go idle, for a time quanta after streaming a time-slice's worth of packets. There is nothing in the above section, or any other section, that teaches any determination of whether an audio command line parameter or audio environment variable is defined. Goetz only teaches exchanging messages between multimedia client and server applications, which include a port number that specifies a port for directing data and a version number that specifies a version of the multimedia client application. Goetz does not teach that any determination made by either the multimedia client or server application as to whether these two variables are defined. To the contrary, the multimedia client application of Goetz sends a "go" message to the multimedia server application. The multimedia server application then verifies receipt of all messages sent, but does not determine whether the variables within the messages are defined. Since Goetz does not teach determining if an audio environment variable or an audio command line parameter is defined, Goetz cannot teach that if the audio environment variable or audio command line parameter is defined, audio data is sent to a platform-independent audio server on a client machine specified by the audio environment variable or audio command line parameter, as recited in claim 4.

Slaughter also does not teach these features. As described above, Slaughter only teaches a platform-independent device driver that requests allocation of memory from a bus manager. Slaughter does not teach or suggest that the platform-independent device driver determines whether an audio command line parameter or audio environment variable is defined. The platform-independent device driver does not perform any other function other than requesting allocation of memory. The platform-independent device driver also does not send or receive audio data.

Therefore, neither Goetz nor Slaughter, either alone or in combination, teach or suggest each and every feature recited in independent claim 4. Thus, in view of the above, Appellant respectfully submits that neither Goetz nor Slaughter, either alone or in combination, teach or suggest each and every feature of independent claim 4 and 9. At least by virtue of their dependency on claims 4 and 9, respectively, neither Goetz nor Slaughter, either alone or in combination, teach or suggest the features of dependent claims 5, 6, 8, 10, 11 and 13. Accordingly, Appellant respectfully requests withdrawal of the rejection of claims 4-6, 8-11 and 13 under 35 U.S.C. § 103(a).

Furthermore, neither Goetz nor Slaughter, either alone or in combination, teaches specific features of dependent claims 4-6, 8-11 and 13. For example, with regard to dependent claim 5, which is representative of claim 10 with regard to similarly recited subject matter, neither Goetz nor Slaughter teach or suggest generating graphic data in a platform-independent application or sending graphic data to a display server on the client machine specified by a display environment variable. The Final Office Action alleges that these features are taught by Goetz at column 11, lines 27-34 and at column 1, lines 61-65, which are reproduced above, and at column 9, lines 31-35 and column 11, lines 2-8, which read as follows:

> For simplicity, the exemplary system **800** creates a file **850** with an audio stream and video stream only, but skilled in the art will appreciate the relevance of the teachings to other media types, e.g., MIDI, etc.
>
> The Web browser application **1010** cooperates with novel multimedia client application **1020** to initiate the processing of file **850**, described above. In turn, multimedia client application **1020** cooperates with novel multimedia server application **1040** to produce the multi media file **850** from the server. The interaction and cooperation of the above entities are further described below.

In the above sections, Goetz only teaches a multimedia server application that generates graphic data (video streams) for a Web browser. Goetz does not teach or suggest a multimedia server application that is platform-independent. As described above, Goetz does not teach or suggest a need exists for a platform-independent multimedia client or server application. Therefore, a person of ordinary skill in the art would not have been motivated to combine or modify the teachings of Goetz and Slaughter to generate graphic data in a platform-independent application.

In addition, neither Goetz nor Slaughter teach or give any incentive to include sending graphic data to a display server specified by a display environment variable. While Goetz teaches a multimedia client application that requests media files, Goetz does not teach that the multimedia client application is specified by a display environment variable. Goetz only teaches sending a message from the multimedia client application to the multimedia server application, which includes a version number that specifies a version of the multimedia client application and a port number that specifies a port to which the multimedia server application directs media files. However, there is no mention of a display environment variable that is specified for a display server on a client machine.

Slaughter also does not teach these features. Slaughter only teaches a platform-independent device driver that requests allocation of memory from a bus manager. The device driver does not generate any graphic data. The device driver also does not send or receive any graphic data. Therefore, a person of ordinary skill in the art would not have been led to combine or modify Goetz to include Slaughter to reach the presently claimed invention, because neither reference teach or suggest a combination of a multimedia client application with a platform-independent device driver. Even if a person of ordinary skill in the art were to combine Goetz and Slaughter's teachings, the result would be a platform-independent device driver that requests allocation of memory in order to send messages with a port number and version number of the multimedia client application to request graphic data. The result would not be sending graphic data to a display server on the client machine specified by a display environment variable, as recited in claim 5.

With regard to claim 6, which is representative of claim 11 with regard to similarly recited subject matter, neither Goetz nor Slaughter teach or suggest that the platform-independent audio server is implemented in the Java programming language. As discussed in the arguments presented above for claims 2 and 15, neither Goetz nor Slaughter teach or suggest a platform-independent audio server, let alone a platform-independent audio server that is implemented in the Java programming language. Therefore, in addition to their dependency on independent claims 4 and 9, neither Goetz nor Slaughter, either alone or in combination, teach or suggest the specific features of claims 6 and 11.

With regard to dependent claim 8, which is representative of claim 13 with regard to similarly recited subject matter, neither Goetz nor Slaughter teach or suggest that the graphic data and the audio data is synchronized. The Final Office Action alleges that Goetz teaches these features at column 1, line 65 to column 2, line 5, which reads as follows:

> For example, to provide intelligible audio-video clips, the video data must be provided to a video driver and audio data must be provided to a sound card driver within specified timing tolerances to maintain intra- and inter-stream synchronism. Intra-stream synchronism means that a given stream, such as audio, is presented in synchronism within specified time relationships, in short, that the stream itself is coherent. Inter-stream synchronism means that multiple related streams are presented in synchronism with respect to each other.

In the above section, Goetz teaches that video data provided to a video driver and audio data provided to the sound card driver should maintain synchronism either with a specific time or between multiple related streams with respect to each other. However, Goetz does not teach that the sound card driver is the same as a platform-independent audio server or that the audio data is sent if an audio environment variable or audio command line parameter is defined. Goetz does not teach or suggest that the sound card driver is a platform-independent audio server. In addition, as discussed in the arguments presented above for claims 4 and 9, there is no determination made in either Goetz or Slaughter as to whether the audio environment variable or an audio command line parameter is defined. There is nothing mentioned in either Goetz or Slaughter about an audio environment variable or an audio command line parameter. Therefore, neither Goetz nor Slaughter teach or suggest that the graphic data and the audio data is synchronized when read in combination with respective independent claims 4 and 9.

Therefore, in addition to their dependency on independent claims 4 and 9, neither Goetz nor Slaughter, either alone or in combination, teaches each and every feature as recited in claim 5, 6, 8, 10, 11 and 13. Thus, Appellant respectfully requests withdrawal of the rejection of dependent claims 4-6, 8-11 and 13 under 35 U.S.C. § 103(a).

II. <u>35 U.S.C. 103(a), Alleged Obviousness, Claims 3, 7, 12 and 16</u>

The Final Office Action rejects claims 3, 7, 12 and 16 under 35 U.S.C. 103(a) as being allegedly unpatentable over Goetz and Slaughter as applied to the claims discussed above, and further in view of Semenzato (U.S. Patent No. 5,903,728). This rejection is respectfully traversed.

As described above, neither Goetz nor Slaughter teach or suggest sending graphic data to a display server on a client machine specified by a display environment variable or sending audio data to an audio server on a client machine specified by an audio environment variable or an audio command line parameter. Semenzato also does not teach these features. Semenzato only teaches a plug-in controller that passes a window identifier to the plug-in body in order display images and information in that window. Semenzato does not teach that the plug-in controller sends graphic data to a display server or audio data to an audio server. In addition, Semenzato only specifies a window identifier for the plug-in controller to access a window, as opposed to a display environment variable for a display server or an audio environment variable/audio command line parameter for an audio server. Therefore, Semenzato also does not teach the features of sending graphic data to a display server on a client machine specified by a display environment variable or sending audio data to an audio server on a client machine specified by an audio environment variable or an audio command line parameter.

As to dependent claim 3, the Final Office Action states:

> Regarding dependent claim 3, Goetz does not specifically teach the display server is an X Windows display server. However, Semenzato taught the display server is an X Windows display server (column 8, lines 56-65). It would have been obvious to one of ordinary skill in the art at the time the invention was made that substituting Semenzato's X Windows display server for Goetz's display server would have been an equivalent substitution. The motivation would have been because X Windows is one many different environments which could implement a distributed multimedia system.

Office Action dated July 29, 2003, pages 6.

Appellant agrees with the Examiner that neither Goetz nor Slaughter teach or suggest that the display server is an X Windows display server. Appellant disagrees with the Examiner that Semenzato teaches these features. Semenzato teaches a plug-in that executes a separate computer process from a platform process in which the plug-in is installed and invokes execution of the plug-in. The platform process creates a window into which the plug-in can display images and information and passes to the plug-in controller a window identifier of the window to thereby grant the plug-in controller access to the window. The plug-in body avoids deletion of the window by creating a sub-window of the window and manipulating the window hierarchy to prevent deletion of the sub-window as a consequence of deletion of the window provided by the platform process (Abstract).

The Final Office Action alleges that the features of claims 3 and 16 are taught by Semenzato at column 8, lines 56-65, which reads as follows:

> Windows are known but are described briefly for completeness. A window resembles a partitioned portion of a computer display and is a virtual display to which

> an individual computer process has exclusive access, i.e., in which the computer process can display graphical data without interference from other computer processes. In some windows-based environments, such as X Windows operating environment of the UNIX operating system, have a window hierarchy in which each window is a sub-window of another window.

Neither the above section nor any other sections of Semenzato teach or suggest any display server contrary to the allegations made by the Final Office Action. In the above section, Semenzato teaches a window hierarchy in which different processes may display graphic data in different windows or sub-windows without interference from other processes. Semenzato also teaches X Windows as an example of an operating environment that supports such window hierarchy in a computer display. However, Semenzato does not teach that the X Windows environment is an X Windows display server running on a client machine, specified by a display environment variable. The X Windows environment of Semenzato is merely an environment that supports a window hierarchy. Nowhere in the reference does Semenzato teach or suggest that the X Windows environment is a display server that is specified by a display environment variable.

Therefore, it would not have been obvious to a person of ordinary skill in the art at the time the invention was made to substitute Semenzato's X Windows display server for Goetz's display server. Goetz teaches a multimedia client application, but does not teach that the multimedia client application is running in an X Windows operating environment. Slaughter teaches a platform-independent device driver that requests allocation of memory, but does not teach that the platform-independent device driver as an X Windows display server. Furthermore, Semenzato does not teach or give any incentive to teach a display server. Semenzato only teaches an X Windows operating environment that supports a window hierarchy, as opposed to a display server. Therefore, neither Goetz, Slaughter nor Semenzato teach or suggest the features of claims 3 and 16.

With regard to dependent claim 7, which is representative of claim 12 with regard to similarly recited subject matter, neither Goetz, Slaughter nor Semenzato teach or suggest that the display server is an X Windows display server. As discussed in the arguments presented above for claims 4 and 9, neither Goetz nor Slaughter teach or suggest determining whether an audio environment variable is defined or an audio command line parameter is defined and if an audio environment variable or an audio command line parameter is defined, sending the audio data to an audio server on a client machine specified by the audio environment variable or by the audio command line parameter. In addition, as described in the arguments presented above for claims 3 and 16, neither

Goetz, Slaughter nor Semenzato teach or suggest that the display server is an X Windows display server.

Thus, in view of the above, Appellant respectfully submits that neither Goetz, Slaughter nor Semenzato, either alone or in combination, teach or suggest each and every feature of dependent claim 3. Since dependent claims 7, 12 and 16 are rejected on the same rationale as claim 3, neither Goetz, Slaughter nor Semenzato teach or suggest each and every feature of claims 7, 12 and 16. Therefore, Appellant respectfully requests withdrawal of the rejection of dependent claims 3, 7, 12 and 16 under 35 U.S.C. § 103(a).

III. Summary

In summary, Goetz, Slaughter and Semenzato fail to teach or suggest the specific features of the present invention. There is nothing in Goetz, Slaughter or Semenzato that teaches or suggests sending graphic data to a display server on a client machine specified by a display environment variable or sending audio data to a platform-independent audio server on the client machine specified by an audio environment variable or by an audio command line parameter. There is also nothing in Goetz, Slaughter or Semenzato that teaches or suggests determining whether an audio environment variable is defined or an audio environment variable or an audio command line parameter is defined. Furthermore, there is nothing in Goetz, Slaughter or Semenzato that teaches or suggests an X Windows display server or a Java audio server.

IV. Conclusion

In view of the above, Appellant respectfully submits that claims 1-16 define over the prior art of record, Goetz, Slaughter and Semenzato. Appellant therefore respectfully requests the Board of Patent Appeals and Interferences to overturn the rejection of claims 1-16 under 35 U.S.C. 103(a).

Respectfully submitted,

[illegible]

Stephen J. Walder, Jr.
Reg. No. 41,534
Carstens, Yee & Cahoon, LLP
PO Box 802334
Dallas, TX 75380
(972) 367-2001

SJW/im

APPENDIX OF CLAIMS

The text of the claims involved in the appeal are:

1. A method for a distributed audio server, the method comprising the computer-implemented steps of:

generating audio data and graphic data in a platform-independent application;

sending the graphic data to a display server on a client machine specified by a display environment variable; and

sending the audio data to a platform-independent audio server on the client machine specified by an audio environment variable or by an audio command line parameter.

2. The method of claim 1 wherein the platform-independent application and the platform-independent audio server are implemented in the Java programming language.

3. The method of claim 1 wherein the display server is an X Windows display server.

4. A method for a distributed audio server, the method comprising the computer-implemented steps of:

generating audio data in a platform-independent application;

in response to receiving the audio data at an audio driver, determining whether an audio environment variable or an audio command line parameter is defined; and

if an audio environment variable or an audio command line parameter is defined, sending the audio data to a platform-independent audio server on a client machine specified by the audio environment variable or by the audio command line parameter.

5. The method of claim 4 further comprising:

generating graphic data in the platform-independent application; and

sending the graphic data to a display server on the client machine specified by a display environment variable.

6. The method of claim 4 wherein the platform-independent application and the platform-independent audio server are implemented in the Java programming language.

7. The method of claim 4 wherein the display server is an X Windows display server.

8. The method of claim 7 wherein the graphic data and the audio data are synchronized.

9. A data processing system for a distributed audio server, the data processing system comprising:

first generating means for generating audio data in a platform-independent application;

determining means for determining, in response to receiving the audio data at an audio driver, whether an audio environment variable or an audio command line parameter is defined; and

first sending means for sending, in response to a determination that an audio environment variable or an audio command line parameter is defined, the audio data to a platform-independent audio server on a client machine specified by the audio environment variable or by the command line parameter.

10. The data processing system of claim 9 further comprising:

second generating means for generating graphic data in the platform-independent application; and

second sending means for sending the graphic data to a display server on the client machine specified by a display environment variable.

11. The data processing system of claim 9 wherein the platform-independent application and the platform-independent audio server are implemented in the Java programming language.

12. The data processing system of claim 9 wherein the display server is an X Windows display server.

13. The data processing system of claim 12 wherein the graphic data and the audio data are synchronized.

14. A computer program product on a computer-readable medium for use in a data processing system for a distributed audio server, the computer program product comprising:

instructions for generating audio data and graphic data in a platform-independent application;

instructions for sending the graphic data to a display server on a client machine specified by a display environment variable; and

instructions sending the audio data to a platform-independent audio server on the client machine specified by an audio environment variable or by an audio command line parameter.

15. The computer program product of claim 14 wherein the platform-independent application and the platform-independent audio server are implemented in the Java programming language.

16. The computer program product of claim 14 wherein the display server is an X Windows display server.